

Cooperative Decentralized Multi-agent Control under Local LTL Tasks and Connectivity Constraints

Meng Guo, Jana Tůmová and Dimos V. Dimarogonas

Abstract—We propose a framework for the decentralized control of a team of agents that are assigned local tasks expressed as Linear Temporal Logic (LTL) formulas. Each local LTL task specification captures both the requirements on the respective agent’s behavior and the requests for the other agents’ collaborations needed to accomplish the task. Furthermore, the agents are subject to communication constraints. The presented solution follows the automata-theoretic approach to LTL model checking, however, it avoids the computationally demanding construction of synchronized product system between the agents. We suggest a decentralized coordination among the agents through a dynamic leader-follower scheme, to guarantee the low-level connectivity maintenance at all times and a progress towards the satisfaction of the leader’s task. By a systematic leader switching, we ensure that each agent’s task will be accomplished.

I. INTRODUCTION

Cooperative control for multi-agent systems have been extensively studied for various purposes like consensus [18], formation [4], [5], and reference-tracking [10], where each agent either serves to accomplish a global objective or fulfil simple local goals such as reachability. In contrast, we focus on planning under complex tasks assigned to the agents, such as periodic surveillance (repeatedly perform A), sequencing (perform A , then B , then C), or request-response (whenever A occurs, perform B). Particularly, we follow the idea of correct-by-design control from temporal logic specifications that has been recently largely investigated both in single-agent and multi-agent settings. In particular, we consider a team of agents modeled as a dynamical system that are assigned a local task specification as Linear Temporal Logic (LTL) formulas. The agents might not be able to accomplish the tasks by themselves and hence requirements on the other agents’ behaviors are also part of the LTL formulas. Consider for instance a team of robot operating in a warehouse that are required to move goods between certain warehouse locations. While light goods can be carried by a single robot, help from another robot is needed to move heavy goods, i.e. the requirement on another agents’ behavior is a part of its LTL task specification.

The goal of this work is to find motion controllers and action plans for the agents that guarantee the satisfaction of all individual LTL tasks. We aim for a decentralized solution while taking into account the constraints that the agents can exchange messages only if they are close enough. Following

the hierarchical approach to LTL planning, we first generate for each agent a sequence of actions as a high-level plan that, if followed, guarantees the accomplishment of the respective agent’s LTL task. Second, we merge and implement the synthesized plans in real-time, upon the run of the system. Namely, we introduce a distributed continuous controller for the leader-follower scheme, where the current leader guides itself and the followers towards the satisfaction of the leader’s task. At the same time, the connectivity of the multi-agent system is maintained. By a systematic leader re-election, we ensure that each agent’s task will be met in long term.

Multi-agent planning under temporal logic tasks has been studied in several recent papers [2], [9], [12], [14]–[16], [19]–[21]. Many of them build on top-down approach to planning, when a single LTL task is given to the whole team. For instance, in [2], [20], the authors propose decomposition of the specification into a conjunction of independent local LTL formulas. On the other hand, we focus on bottom-up planning from individual specification. Related work includes a decentralized control of a robotic team from local LTL specification with communication constraints proposed in [6]. However, the specifications there are truly local and the agents do not impose any requirements on the other agents’ behavior. In [9], the same bottom-up planning problem from LTL specifications is considered and a partially decentralized solution is designed that takes into account only clusters of dependent agents instead of the whole group. This approach is later extended in [19], where a receding horizon approach to the problem is suggested. Both mentioned studies however assume that the agents are fully synchronized in their discrete abstractions and the proposed solutions rely on construction of the synchronized product system between the agents, or at least of its part. In contrast, in this work, we avoid the product construction completely.

The contribution of the paper can be summarized as the proposal of a decentralized motion and action control scheme for multi-agent systems with complex local tasks which handles both connectivity constraints and collaborative tasks. The features of the suggested solution are as follows: (1) the continuous controller is distributed and integrated with the leader election scheme; (2) the distributed leader election algorithm only requires local communications and guarantees sequential progresses towards individual desired tasks; and (3) the proposed coordination scheme operates in real-time, upon the run of the system as opposed to offline solutions that require fully synchronized motions of all agents.

The rest of the paper is organized as follows. In Section II we state the necessary preliminaries. Section III formally

The authors are with the ACCESS Linnaeus Center, School of Electrical Engineering, KTH Royal Institute of Technology, SE-100 44, Stockholm, Sweden and with the KTH Centre for Autonomous Systems. mengg, tumova, dimos@kth.se. This work was supported by the EU STREP RECONFIG: FP7-ICT-2011-9-600825.

introduces the considered problem. In Section IV we describe the proposed solution in details. Section V demonstrates the results in a simulated case study. Finally, we conclude in Section VI.

II. PRELIMINARIES

Given a set S , let 2^S , and S^ω denote the set of all subsets of S , and the set of all infinite sequences of elements of S , respectively. An infinite sequence of elements of S is called an infinite word over S , respectively.

Definition 1 An LTL formula ϕ over the set of services Σ is defined inductively as follows:

- 1) every service $\sigma \in \Sigma$ is a formula, and
- 2) if ϕ_1 and ϕ_2 are formulas, then $\phi_1 \vee \phi_2$, $\neg \phi_1$, $X \phi_1$, $\phi_1 U \phi_2$, $F \phi_1$, and $G \phi_1$ are each formulas,

where \neg (negation) and \vee (disjunction) are standard Boolean connectives, and X (next), U (until), F (eventually), and G (always) are temporal operators.

The semantics of LTL is defined over infinite words over 2^Σ . Intuitively, σ is satisfied on a word $w = w(1)w(2)\dots$ if it holds at its first position $w(1)$, i.e. if $\sigma \in w(1)$. Formula $X\phi$ holds true if ϕ is satisfied on the word suffix that begins in the next position $w(2)$, whereas $\phi_1 U \phi_2$ states that ϕ_1 has to be true until ϕ_2 becomes true. Finally, $F\phi$ and $G\phi$ are true if ϕ holds on w eventually, and always, respectively. For the formal definition of the LTL semantics see, e.g. [1].

The set of all words that are accepted by an LTL formula ϕ is denoted by $\mathcal{L}(\phi)$.

Definition 2 (Büchi Automaton) A Büchi automaton over alphabet 2^Σ is a tuple $\mathcal{B} = (Q, q_{init}, 2^\Sigma, \delta, F)$, where

- Q is a finite set of states;
- $q_{init} \in Q$ is the initial state;
- 2^Σ is an input alphabet;
- $\delta \subseteq Q \times \Sigma \times Q$ is a non-deterministic transition relation;
- F is the acceptance condition.

The semantics of Büchi automata are defined over infinite input words over 2^Σ . A run of the Büchi automaton \mathcal{B} over an input word $w = w(1)w(2)\dots$ is a sequence $\rho = q_1q_2\dots$, such that $q_1 = q_{init}$, and $(q_i, w(i), q_{i+1}) \in \delta$, for all $i \geq 1$. A run $\rho = q_1q_2\dots$ is *accepting* if it intersects F infinitely many times. A word w is *accepted* by \mathcal{B} if there exists an accepting run over w . The *language* of all words accepted by \mathcal{B} is denoted by $\mathcal{L}(\mathcal{B})$. Any LTL formula ϕ over Π can be algorithmically translated into a Büchi automaton \mathcal{B} , such that $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\phi)$ [1] and many software tools for the translation exist, e.g., [8].

Given an LTL formula φ over Σ , a word that satisfies φ can be generated as follows. First, the LTL formula is translated into a corresponding Büchi automaton. Second, the Büchi automaton is viewed as a graph $G = (V, E)$, where $V = Q$, and E is given by the transition relation δ in the expected way: $(q, q') \in E \iff \exists S \subseteq \Sigma$, such that

$(q, S, q') \in \delta$. By finding a finite path (prefix) followed by a cycle (suffix) containing an accepting state, we find a word that is accepted by \mathcal{B} , which is a word that satisfies φ in a prefix-suffix form $S_1 \dots S_p(S_{p+1} \dots S_s)^\omega$. Details can be found e.g., in [1].

In this particular work, we are interested only in subsets of 2^Σ that are singletons. Thus, with a slight abuse of notation, we interpret LTL over words over Σ , i.e. over sequences of services instead of sequences of subsets of services.

III. PROBLEM FORMULATION

A. Agent Dynamics and Network Structure

Let us consider a team of N agents, modeled by the single-integrator dynamics:

$$\dot{x}_i(t) = u_i(t), \quad i \in \mathcal{N} = \{1, \dots, N\}, \quad (1)$$

where $x_i(t)$, $u_i(t) \in \mathbb{R}^2$ are the state and control inputs of agent i at time $t > 0$, $x_i(0)$ is the given initial state, and $x_i(t)$ is the *trajectory* of agent i from 0 to $t \geq 0$. We assume that all agents start at the same instant $t = 0$.

Suppose that each of the agents has a limited communication radius of $r > 0$. This means that at time t , agent i can communicate, i.e., exchange messages directly with agent j if and only if $\|x_i(t) - x_j(t)\| \leq r$. This constraint imposes certain challenges on the distributed coordination of multi-agent systems as the inter-agent communication or information exchange depends on their relative positions.

Agents i and j are *connected* at time t if and only if either $\|x_i(t) - x_j(t)\| \leq r$, or if there exists i' , such that $\|x_i(t) - x_{i'}(t)\| \leq r$, where i' and j are connected. Hence, two connected agents can communicate indirectly. We assume that initially, all agents are connected. The particular message passing protocol is beyond the scope of this paper. For simplicity, we assume that message delivery is reliable, meaning that a message sent by agent i will be received by all connected agents j .

B. Task Specifications

Each agent $i \in \mathcal{N}$ is assigned a set of M_i services $\Sigma_i = \{\sigma_{ih}, h \in \{1, \dots, M_i\}\}$ that it is responsible for, and a set of K_i regions, where subsets of these services can be *provided*, denoted by $\mathcal{R}_i = \{R_{ig}, g \in \{1, \dots, K_i\}\}$. For simplicity of presentation, R_{ig} is determined by a circular area:

$$R_{ig} = \{y \in \mathbb{R}^2 \mid \|y - c_{ig}\| \leq r_{ig}\} \quad (2)$$

where $c_{ig} \in \mathbb{R}^2$ and r_{ig} are the center and radius of the region, respectively, such that $r_{ig} \geq r_{min} > 0$, for a fixed minimal radius r_{min} . Furthermore, each region in \mathcal{R}_i is reachable for each agent. Labeling function $L_i : \mathcal{R}_i \rightarrow 2^{\Sigma_i}$ assigns to each region R_{ig} the set of services $L_i(R_{ig}) \subseteq \Sigma_i$ that can be provided in there.

Some of the services in Σ_i can be provided solely by the agent i , while others require cooperation with some other agents. Formally, agent i is associated with a set of *actions* Π_i that it is capable of *executing*. The actions are of two types:

- action π_{ih} of *providing* the service $\sigma_{ih} \in \Sigma_i$; and

- action $\varpi_{i'i'h'}$ of *cooperating* with the agent i' in providing its service $\sigma_{i'h'} \in \Sigma_{i'}$.

A service σ_{ih} then takes the following form:

$$\sigma_{ih} = \pi_{ih} \wedge \bigwedge_{i' \in \mathcal{C}_{ih}} \varpi_{i'ih}, \quad (3)$$

for the set of cooperating agents \mathcal{C}_{ih} , where $\emptyset \subseteq \mathcal{C}_{ih} \subseteq \mathcal{N} \setminus \{i\}$. Informally, a service σ_i is provided if the agent's relevant service-providing action and the corresponding co-operating agents' actions are executed at the same time. Furthermore, it is required that at the moment of service providing, the agent and the cooperating agents from \mathcal{C}_{ih} occupy the same region R_{ig} , where $\sigma_{ih} \in L_i(R_{ig})$.

Definition 3 (Trace) A valid trace of agent i is a tuple $trace_i = (\mathbf{x}_i(t), \mathbb{T}_i^A, \mathbb{A}_i, \mathbb{T}_i^S, \mathbb{S}_i)$, where

- $\mathbf{x}_i(t)$ is a trajectory of agent i ;
- $\mathbb{T}_i^A = t_1, t_2, t_3, \dots$ is the sequence of time instances when agent i executes actions from Π_i ;
- $\mathbb{A}_i : \mathbb{T}_i^A \rightarrow \Pi_i$ represents the sequence of executed actions, both the service-providing and the cooperating ones;
- $\mathbb{T}_i^S = \tau_1, \tau_2, \tau_3, \dots$ is a sequence of time instances when services from Σ_i are provided. Note that \mathbb{T}_i^S is a subsequence of \mathbb{T}_i^A and it is equal to the time instances when service-providing actions are executed; and
- $\mathbb{S}_i : \mathbb{T}_i^S \rightarrow \Sigma_i$ represents the sequence of provided services that satisfies the following property for all $l \geq 1$: There exists $g \in \{1, \dots, K_i\}$, such that
 - $\mathbf{x}_i(\tau_l) \in R_{ig}$, $\mathbb{S}_i(\tau_l) \in L_i(R_{ig})$, and $\mathbb{S}_i(\tau_l) = \sigma_{ih} \Rightarrow \mathbb{A}_i(\tau_l) = \pi_{ih}$, and
 - for all $i' \in \mathcal{C}_{ih}$, it holds that $\mathbf{x}_{i'}(\tau_l) \in R_{ig}$ and $\mathbb{A}_{i'}(\tau_l) = \varpi_{i'ih}$.

In other words, the agent i can provide a service σ_{ih} only if (i) it is present in a region R_{ig} , where this service can be provided, and it executes the relevant service-providing action π_{ih} itself, and (ii) all its cooperating agents from \mathcal{C}_{ih} are present in the same region R_{ig} as agent i and execute the respective cooperative actions needed.

Definition 4 (LTL Satisfaction) A valid trace $trace_i = (\mathbf{x}_i(t), \mathbb{T}_i^A, \mathbb{A}_i, \mathbb{T}_i^S = \tau_1, \tau_2, \tau_3, \dots, \mathbb{S}_i : \mathbb{T}_i^S \rightarrow \Sigma_i)$, satisfies an LTL formula over φ_i , denoted by $trace_i \models \varphi_i$ if and only if $\mathbb{S}_i(\tau_1)\mathbb{S}_i(\tau_2)\mathbb{S}_i(\tau_3) \dots \models \varphi_i$.

Remark 1 Traditionally, LTL is defined over the set of atomic propositions (APs) instead of services (see, e.g. [1]). Usually APs represent inherent properties of system states. The labeling function L then partitions APs into those that are true and false in each state. The LTL formulas are interpreted over trajectories of systems or their discrete abstractions.

In this work, we consider an alternative definition of LTL semantics to describe the desired tasks. Particularly, we perceive atomic propositions as offered services rather than undetachable inherent properties of the system states. For

instance, given that a state is determined by the physical location of an agent, we consider atomic propositions of form “in this location, an object can be loaded”, or “there is a recharger in this location” rather than “this location is dangerous”. In other words, the agent is in our case given the option to decide whether an atomic proposition $\sigma_{ih} \in L(R_{ig})$ is in state $x_i(t) \in R_{ig}$ satisfied or not. In contrast, $\sigma_i \in \Sigma_i$ is never satisfied in state $x_i(t) \in R_{ig}$, such that $\sigma_i \notin L(R_{ig})$. The LTL specifications are thus interpreted over the sequences of provided services along the trajectories instead of the trajectories themselves.

C. Problem statement

Given the above settings, we now formally state our problem:

Problem 1 Given a team of the agents \mathcal{N} subject to dynamics in Eq. 1, synthesize for each agent $i \in \mathcal{N}$

- a control input u_i
- a time sequence \mathbb{T}_i^A , and
- an action sequence \mathbb{A}_i ,

such that the trace $trace_i = (\mathbf{x}_i(t), \mathbb{T}_i^A, \mathbb{T}_i^S, \mathbb{A}_i, \mathbb{S}_i)$ is valid and satisfies the given local LTL task specification φ_i over the set of services Σ_i .

IV. PROBLEM SOLUTION

Our approach to the problem involves an offline and an online step. In the offline step, we synthesize a high-level plan in the form of a sequence of services for each of the agents. In the online step, we dynamically switch between the high-level plans through leader election. The whole team then follows the leader towards providing its next service.

In this section, we provide the details of the proposed solution. Namely, we define the notion of connectivity graph for the multi-agent system as a necessary condition for the rest of the solution. Further, we focus on decentralized control of the whole team of agents towards a selected goal region $R_{\ell g}$ that is known only to a leading agent ℓ while maintaining their connectivity. Finally, we discuss the election of leading agents and progressive services to be provided, and goal regions to be visited that guarantee the satisfaction of all agents' tasks in long term.

A. Connectivity Graph

Before discussing the structure of the proposed solution, let us introduce the notion of agents' connectivity graph that will allow us to handle the constraints imposed on communication between the agents.

Recall that each agent has a limited communication radius $r > 0$ as defined in Section III-A. Moreover, let $\varepsilon \in (0, r)$ be a given constant. It is worth mentioning that ε plays an important role for the edge definition below. In particular, it introduces a hysteresis in the definition for adding new edges to the communication graph.

Definition 5 Let $G(t) = (\mathcal{N}, E(t))$ denote the undirected time-varying connectivity graph formed by the agents, where

$E(t) \subseteq \mathcal{N} \times \mathcal{N}$ is the edge set for $t \geq 0$. At time $t = 0$, we set $E(0) = \{(i, j) \mid \|x_i(0) - x_j(0)\| < r\}$. At time $t > 0$, $(i, j) \in E(t)$ if and only if one of the following conditions hold:

- (i) $\|x_i(t) - x_j(t)\| \leq r - \varepsilon$, or
- (ii) $r - \varepsilon < \|x_i(t) - x_j(t)\| \leq r$ and $(i, j) \in E(t^-)$, where $t^- < t$ and $|t - t^-| \rightarrow 0$.

Note that the condition (ii) in the above definition guarantees that a new edge will only be added when the distance between two unconnected agents decreases below $r - \varepsilon$. This property is crucial in proving the connectivity maintenance by Lemma 1 and the convergence by Lemma 2.

Consequently, each agent $i \in \mathcal{N}$ has a time-varying set of neighbouring agents, with which it can communicate directly, denoted by $\mathcal{N}_i(t) = \{i' \in \mathcal{N} \mid (i, i') \in E(t)\}$. Note that if j is reachable from i in $G(t)$ then agents i and j are connected, i.e., they can communicate directly or indirectly. From the initial connectivity requirement, we have that $G(0)$ is connected. Hence, maintaining $G(t)$ connected for all $t \geq 0$ ensures that the agents are always connected, too.

B. Continuous Controller Design

In this section, let us firstly focus on the following problem: given a leader $\ell \in \mathcal{N}$ at time t and a goal region $R_{\ell g} \in \mathcal{R}_{\ell}$, propose a decentralized continuous controller that: (1) guarantees that all agents $i \in \mathcal{N}$ reach $R_{\ell g}$ at a finite time $\bar{t} < \infty$; (2) $G(t')$ remains connected for all $t' \in [t, \bar{t}]$. Both objectives are critical for the leader selection scheme introduced in Section IV-C, which ensures sequential satisfaction of φ_i for each $i \in \mathcal{N}$.

Denote by $x_{ij}(t) = x_i(t) - x_j(t)$ the pairwise relative position between neighbouring agents, $\forall (i, j) \in E(t)$. Thus $\|x_{ij}(t)\|^2 = (x_i(t) - x_j(t))^T (x_i(t) - x_j(t))$ denotes the corresponding distance. We propose the continuous controller with the following structure:

$$u_i(t) = -b_i(x_i - c_{ig}) - \sum_{j \in \mathcal{N}_i(t)} \nabla_{x_i} \phi(\|x_{ij}\|), \quad (4)$$

where $\nabla_{x_i} \phi(\cdot)$ is the gradient of the potential function $\phi(\|x_{ij}\|)$ with respect to x_i , which is to be defined; $b_i \in \{0, 1\}$ indicates if agent i is the leader; $c_{ig} \in \mathbb{R}^2$ is the center of the next goal region for agent i ; b_i and c_{ig} are derived from the leader selection scheme in Section IV-C later.

The potential function $\phi(\|x_{ij}\|)$ is defined as follows

$$\phi(\|x_{ij}\|) = \frac{\|x_{ij}\|^2}{r^2 - \|x_{ij}\|^2}, \quad \|x_{ij}\| \in [0, r), \quad (5)$$

and has the following properties: (1) its partial derivative of $\phi(\cdot)$ over $\|x_{ij}\|$ is given by

$$\frac{\partial \phi(\|x_{ij}\|)}{\partial \|x_{ij}\|} = \frac{-2r^2 \|x_{ij}\|}{(r^2 - \|x_{ij}\|^2)^2} \geq 0 \quad (6)$$

for $\|x_{ij}(t)\| \in [0, r)$ and the equality holds when $\|x_{ij}\| = 0$; (2) $\phi(\|x_{ij}\|) \rightarrow 0$ when $\|x_{ij}\| \rightarrow 0$; (3) $\phi(\|x_{ij}\|) \rightarrow +\infty$

when $\|x_{ij}\| \in [0, r)$. As a result, controller (4) becomes

$$u_i(t) = -b_i(x_i - c_{ig}) - \sum_{j \in \mathcal{N}_i(t)} \frac{2r^2}{(r^2 - \|x_{ij}\|^2)^2} (x_i - x_j), \quad (7)$$

which is fully distributed as it only depends x_i and x_j , $\forall j \in \mathcal{N}_i(t)$.

Lemma 1 Assume that $G(t)$ is connected at $t = T_1$ and agent $\ell \in \mathcal{N}$ is the fixed leader for all $t \geq T_1$. By applying the controller in Eq. (7), $G(t)$ remains connected and $E(T_1) \subseteq E(t)$ for $t \geq T_1$.

Proof: Assume that $G(t)$ remains invariant during $[t_1, t_2) \subseteq [T_1, \infty)$, i.e., no new edges are added to $G(t)$. Consider the following function:

$$V(t) = \frac{1}{2} \sum_{i=1}^N \sum_{j \in \mathcal{N}_i(t)} \phi(\|x_{ij}\|) + \frac{1}{2} \sum_{i=1}^N b_i (x_i - c_{ig})^T (x_i - c_{ig}), \quad (8)$$

which is positive semi-definite. The time derivative of (8) along system (1) is given by

$$\begin{aligned} \dot{V}(t) &= \sum_{i=1}^N \frac{\partial V}{\partial x_i} \dot{x}_i \\ &= \sum_{i=1, i \neq \ell}^N \left(\left(\sum_{j \in \mathcal{N}_i(t)} \nabla_{x_i} \phi(\|x_{ij}\|) \right) u_i \right) \\ &\quad + \left(\sum_{j \in \mathcal{N}_\ell(t)} \nabla_{x_\ell} \phi(\|x_{\ell j}\|) + (x_\ell - c_{\ell g}) \right) u_\ell. \end{aligned} \quad (9)$$

By (4), for follower $i \neq \ell$, the control input is given by

$$u_i = - \sum_{j \in \mathcal{N}_i(t)} \nabla_{x_i} \phi(\|x_{ij}\|)$$

since $b_i = 0$ for all followers. For the single leader ℓ , its control input is given by

$$u_\ell = -(x_\ell - c_{\ell g}) - \sum_{j \in \mathcal{N}_\ell(t)} \nabla_{x_\ell} \phi(\|x_{\ell j}\|)$$

since $b_\ell = 1$. This implies that

$$\begin{aligned} \dot{V}(t) &= - \sum_{i=1, i \neq \ell}^N \left\| \sum_{j \in \mathcal{N}_i(t)} \nabla_{x_i} \phi(\|x_{ij}\|) \right\|^2 \\ &\quad - \left\| (x_\ell - c_{\ell g}) + \sum_{j \in \mathcal{N}_\ell(t)} \nabla_{x_\ell} \phi(\|x_{\ell j}\|) \right\|^2 \leq 0. \end{aligned} \quad (10)$$

Thus $V(t) \leq V(0) < +\infty$ for $t \in [t_1, t_2)$. It means that during $[t_1, t_2)$, no existing edge can have a length close to r , i.e., no existing edge will be *lost* by the definition of an edge.

On the other hand, assume a *new* edge (p, q) is added to $G(t)$ at $t = t_2$, where $p, q \in \mathcal{N}$. By Definition 5, it holds that $\|x_{pq}(t_2)\| \leq r - \varepsilon$ and $\phi(\|x_{pq}(t_2)\|) = \frac{r - \varepsilon}{\varepsilon(2r - \varepsilon)} < +\infty$ since $0 < \varepsilon < r$. Denote the set of newly-added edges at $t = t_2$ as $\hat{E} \subset \mathcal{N} \times \mathcal{N}$. Let $V(t_2^+)$ and $V(t_2^-)$ be the value

of Lyapunov function from (8) before and after adding the set of new edges to $G(t)$ at $t = t_2$. We get

$$\begin{aligned} V(t_2^+) &= V(t_2^-) + \sum_{(p,q) \in \widehat{E}} \phi(\|x_{pq}(t_2)\|) \\ &\leq V(t_2^-) + |\widehat{E}| \frac{r - \varepsilon}{\varepsilon(2r - \varepsilon)} < +\infty. \end{aligned} \quad (11)$$

Thus $V(t) < \infty$ also holds when new edges are added. As a result, $V(t) < +\infty$ for $t \in [T_1, \infty)$. By Definition 5, one existing edge $(i, j) \in E(t)$ will be lost only if $x_{ij}(t) = r$. It implies that $\phi(\|x_{ij}\|) \rightarrow +\infty$, i.e., $V(t) \rightarrow +\infty$ by (8). By contradiction, we can conclude that new edges might be added but no existing edges will be lost, namely $E(T_1) \subseteq E(t)$, $\forall t \geq T_1$.

To conclude, given a connected $G(t)$ at $t = T_1$ and a fixed leader $\ell \in \mathcal{N}$ for $t \geq T_1$, it is guaranteed that $G(t)$ remains connected, $\forall t \geq T_1$. ■

Lemma 2 *Given that $G(t)$ is connected at $t = T_1$ and the fixed leader $\ell \in \mathcal{N}$ for $t \geq T_1$, it is guaranteed that under controller in Eq. (7) there exist $T_1 \leq \bar{t} < +\infty$*

$$x_i(\bar{t}) \in R_{\ell g}, \quad \forall i \in \mathcal{N}. \quad (12)$$

Proof: First of all, it is shown in Lemma 1 that $G(t)$ remains connected for $t \geq T_1$ if $G(T_1)$ is connected. Moreover $E(T_1) \subseteq E(t)$, $\forall t \geq T_1$, i.e., no existing edges will be lost.

Now we show that all agents converge to the goal region of the leader in finite time. By (10), $\dot{V}(t) \leq 0$ for $t \geq T_1$ and $\dot{V}(t) = 0$ when the following conditions hold: (1) for $i \neq \ell$ and $i \in \mathcal{N}$, it holds that

$$\sum_{j \in \mathcal{N}_i(t)} \frac{2r^2}{(r^2 - \|x_{ij}\|^2)^2} (x_i - x_j) = 0; \quad (13)$$

(2) for the leader $\ell \in \mathcal{N}$, it holds that

$$(x_\ell - c_{\ell g}) + \sum_{j \in \mathcal{N}_\ell(t)} \frac{2r^2}{(r^2 - \|x_{\ell j}\|^2)^2} (x_\ell - x_j) = 0. \quad (14)$$

Denote by

$$h_{ij} = \frac{2r^2}{(r^2 - \|x_{ij}\|^2)^2}, \quad \forall (i, j) \in E(t). \quad (15)$$

We can construct a $N \times N$ matrix H satisfying $H(i, i) = \sum_{j \in \mathcal{N}_i} h_{ij}$ and $H(i, j) = -h_{ij}$, where $i \neq j \in \mathcal{N}$. Since $x_{ij} \in [0, r - \varepsilon]$, $\forall (i, j) \in E(t)$, it holds that $h_{ij} > 0$. As shown in [17], H is positive semidefinite with a single eigenvalue at the origin, of which the corresponding eigenvector is the unit column vector of length N , denoted by $\mathbf{1}_N$. By combining (13) and (14), we get

$$H \otimes I_2 \cdot \mathbf{x} + (\mathbf{x} - \mathbf{c}) = 0 \quad (16)$$

where \otimes denotes the Kronecker product [11]; \mathbf{x} is the stack vector for x_i , $i \in \mathcal{N}$; I_2 is the 2×2 identity matrix; $\mathbf{c} = \mathbf{1}_N \otimes c_{\ell g}$. Then

$$H \otimes I_2 \cdot \mathbf{c} = (H \otimes I_2) \cdot (\mathbf{1}_N \otimes c_{\ell g}) = (H \cdot \mathbf{1}_N) \otimes (I_2 \cdot c_{\ell g}).$$

Since $H \cdot \mathbf{1}_N = \mathbf{0}_N$, it implies that $H \otimes I_2 \cdot \mathbf{c} = \mathbf{0}_{2N}$. By (16), it implies that $H \otimes I_2 \cdot (\mathbf{x} - \mathbf{c}) = \mathbf{0}$. Since we have shown that H is positive semidefinite with one eigenvalue at the origin, (16) holds only when $\mathbf{x} = \mathbf{c}$, i.e., $x_i = c_{\ell g}$, $\forall i \in \mathcal{N}$.

By LaSalle's Invariance principle [13], the closed-loop system under controller in Eq. (7) will converge to the largest invariant set inside the region

$$S = \{\mathbf{x} \in \mathbb{R}^{2N} \mid x_i = c_{\ell g}, \forall i \in \mathcal{N}\}, \quad (17)$$

as $t \rightarrow +\infty$. In other words, it means that all agents in \mathcal{N} converge to the same point $c_{\ell g}$. Since clearly $c_{\ell g} \in R_{\ell g}$, by continuity all agents would enter $R_{\ell g}$ which has a minimal radius r_{\min} by (2). Consequently, there exists $\bar{t} < +\infty$ that $x_i(\bar{t}) \in R_{\ell g}$, $\forall i \in \mathcal{N}$.

To conclude, given a connected initial graph $G(T_1)$ and the fixed leader $\ell \in \mathcal{N}$ for $t \geq T_1$, it is guaranteed that under controller in Eq. (7) all agents will converge to the region $R_{\ell g}$ in finite time. ■

C. Progressive Goal and Leader Election

To complete the solution to Problem 1, we discuss the election of the leader ℓ and the choice of a goal region $R_{\ell g}$ at time t . As the first offline and fully decentralized step, we generate for each agent i a *high-level plan*, which is represented by the sequence of services that, if provided, guarantee the satisfaction of φ_i . Secondly, in a repetitive online procedure, each agent i is assigned a value that, intuitively, represents the agent's urge to provide the next service in its high-level plan. Using ideas from bully leader election algorithm [7], an agent with the strongest urge is always elected as a leader within the connectivity graph. By changing the urge dynamically at the times when services are provided, we ensure that each of the agents is elected as a leader infinitely often. Thus, each agent's precomputed high-level plan is followed.

1) *Offline high-level plan computation:* Given an agent $i \in \mathcal{N}$, a set of services Σ_i , and an LTL formula φ_i over Σ_i , a high-level plan for i can be computed via standard model-checking methods as described in Section II. Roughly, by translating φ_i into a language equivalent Büchi automaton and by consecutive analysis of the automaton, a sequence of services $\Omega_i = \sigma_{i1} \dots \sigma_{ip_i}(\sigma_{ip_i+1} \dots \sigma_{is_i})^\omega$, such that $\Omega_i \models \varphi_i$ can be found.

2) *Urge function:* Let i be a fixed agent, t the current time and $\sigma_{i1} \dots \sigma_{ik}$ a prefix of services of the high-level plan Ω_i that have been provided till t . Moreover, let $\tau_{i\lambda}$ denote the time, when the latest service, i.e., $\sigma_{i\lambda} = \sigma_{ik}$ was provided, or $\tau_{i\lambda} = 0$ in case no service prefix of Ω_i has been provided, yet.

Using $\tau_{i\lambda}$, we could define agent i 's *urge* at time t as a tuple

$$\Upsilon_i(t) = (t - \tau_{i\lambda}, i). \quad (18)$$

Furthermore, to compare the agents' urges at time t , we use lexicographical ordering: $\Upsilon_i(t) > \Upsilon_j(t)$ if and only if

- $t - \tau_{i\lambda} > t - \tau_{j\lambda}$, or
- $t - \tau_{i\lambda} = t - \tau_{j\lambda}$, and $i > j$.

Note that $i \neq j$ implies that $\Upsilon_i(t) \neq \Upsilon_j(t)$, for all $t \geq 0$. As a result, the defined ordering is a linear ordering and at any time t , there exists exactly one agent i maximizing its urge $\Upsilon_i(t)$.

3) *Overall algorithm:* The algorithm for an agent $i \in \mathcal{N}$ is summarized in Alg. 1 and is run on each agent separately, starting at time $t = 0$.

Algorithm 1 Solution to Prob. 1

Input: Agents' own ID i , the set of all agent IDs \mathcal{N} , formula φ_i
Output: $trace_i$

```

1: compute plan  $\Omega_i := \sigma_{i1} \dots \sigma_{ip_i}(\sigma_{ip_i+1} \dots \sigma_{is_i})^\omega$ 
2:  $\tau_{i\lambda} := 0$ ;  $\sigma_{i\nu} := \sigma_{i1}$ 
3: send ready( $i$ ) and wait to receive ready( $j$ ) for all  $j \in \mathcal{N} \setminus \{i\}$ 

4: if  $i = N$  then
5:   send init_elect( $i, t_{curr}$ ), where  $t_{curr}$  is the current time
6: end if
7: loop
8:   wait to receive a message  $m$ 
9:   switch  $m$ 
10:    case  $m = \text{init\_elect}(i', t)$  for some  $i' \in \mathcal{N}$  and time  $t$ 
11:      send  $\text{me}(\Upsilon_i(t))$  and wait to receive  $\text{me}(\Upsilon_j(t))$  from all  $j \in \mathcal{N} \setminus \{i\}$ 
12:      elect the leader  $\ell \in \mathcal{N}$  maximizing  $\Upsilon_\ell(t)$ 
13:      send finish_elect( $i$ ) and wait to receive finish_elect( $j$ ) from all  $j \in \mathcal{N} \setminus \{i\}$ 
14:      if  $\ell = i$  then
15:         $b_i := 1$ 
16:        pick  $R_{\ell g} = R_{ig}$ , such that  $\sigma_{i\nu} \in L_i(R_{ig})$ 
17:        repeat
18:          apply controller  $u_i$  from Eq. (7)
19:        until  $x_j(t) \in R_{\ell g}$  for all  $j \in \{i\} \cup \mathcal{C}_{i\nu}$ 
20:        send execute_request( $\varpi_{j i\nu}$ ) for all  $j \in \mathcal{C}_{i\nu}$ 
21:        execute  $\pi_{i\nu}$ 
22:         $\tau_{i\lambda} := 0$ ;  $\sigma_{i\nu} := \sigma_{i\nu+1}$ 
23:        update prefixes of  $\mathbb{T}_i^A, \mathbb{A}_i, \mathbb{T}_i^S$ , and  $\mathbb{S}_i$ 
24:        send init_elect( $i, t_{curr}$ ), where  $t_{curr}$  is the current time
15:      else
26:         $b_i := 0$ 
27:        repeat
28:          apply controller  $u_i$  from Eq. (7)
29:        until a message  $m$  is received; goto line 9
30:      end if
31:      case  $m = \text{execute\_request}(\varpi_{ii'h'})$  for some  $i' \in \mathcal{N}$ , and  $\sigma_{i'h'} \in \Sigma_{i'}$ 
32:        execute  $\varpi_{ii'h'}$ 
33:        update prefixes of  $\mathbb{T}_i^A$ , and  $\mathbb{A}_i$ ; goto line 9
34:      end switch
35: end loop

```

The algorithm is initialized with the offline computation of the high-level plan $\Omega_i = \sigma_{i1} \dots \sigma_{ip_i}(\sigma_{ip_i+1} \dots \sigma_{is_i})^\omega$ as outlined above, and setting the values $\tau_{i\lambda} = 0$, $\sigma_{i\nu} = \sigma_{i1}$ (lines 1 – 2). Then, the agent broadcasts a message to acknowledge the others that it is ready to proceed and waits to receive analogous messages from the remaining agents (line 3). The first leader election is triggered by a message sent by the agent N (lines 4 – 6) equipped with the time stamp t_{curr} of the current time.

Several types of messages can be received by the agent i . Message $\text{init_elect}(i', t)$, where i' is an arbitrary agent ID

and t is a time stamp, notifies that leader re-election is triggered (line 10). In such a case, the agent sends out the message $\text{me}(\Upsilon_i(t))$ containing its own urge value $\Upsilon_i(t)$ at the received time t and waits to receive analogous messages from the others (line 11). The agent with the maximal urge is elected as the leader (line 12) and the algorithm proceeds when each of the agents has set the new leader (line 13). Note that the elected leader is the same for all the agents.

The rest of the algorithm differs depending on whether the agent i is the leader (14–25) or not (25–30). The leader applies the controller from Eq. (7) to reach a region where the next service $\sigma_{i\nu}$ can be provided (lines 15–19). At the same time, it waits for the cooperating agents to reach the same region (line 19). Then it provides service $\sigma_{i\nu}$, with the help of the others (lines 20–21) and it sets the new latest service providing time $\tau_{i\lambda} = 0$, and the next service to be provided $\sigma_{i\nu}$ to the following service in its plan, i.e. $\sigma_{i\nu+1}$, where, with a slight abuse of notation, we assume that $\sigma_{is_i+1} = \sigma_{ip_i+1}$ (line 22). For simplicity of presentation, we assume that the execution of an action $\pi_{i\nu}$ is synchronized with the execution of the action $\varpi_{j i\nu}$, for all $j \in \mathcal{C}_{i\nu}$. The details of the synchronization procedure are beyond the scope of this paper; for instance, the leader can decide a future time instance when $\varpi_{j i\nu}$ should be executed and send it as a part of the execute_request message. Finally, the leader triggers a leader re-election (line 24) with the current time t_{curr} as a time stamp.

A follower simply applies the controller from Eq. (7) until it receives a message from the leader (lines 25–29). The message can be either $\text{execute_request}(\varpi_{ii'h'})$ for a cooperating agent or $\text{init_elect}(i', t)$ for a non-cooperating agent.

The algorithm naturally determines the trace $trace_i = (\mathbf{x}_i(t), \mathbb{T}_i^A, \mathbb{A}_i, \mathbb{T}_i^S, \mathbb{S}_i)$ of the agent i as follows: The trajectory $\mathbf{x}_i(t)$ is given through the application of the controller u_i from Eq. (7) (lines 18 and 28). The sequences $\mathbb{T}_i^A, \mathbb{A}_i, \mathbb{T}_i^S, \mathbb{S}_i$ are iteratively updated upon the agent's run (lines 23 and 33). Initially, they are all empty sequences and a time instant, an action, or a service is added to them whenever an action is executed (lines 21 and 32) or a service is provided (line 21), respectively.

To prove that the proposed algorithm is correct, we first prove that each agent i is elected as a leader infinitely many times:

Lemma 3 *Given an agent $i \in \mathcal{N}$ at time t , there exists $T \geq t$, such that $\Upsilon_i(T) > \Upsilon_j(T)$, for all $j \in \mathcal{N}$, and $t \geq 0$.*

Proof: Proof is given by contradiction. Assume the following:

Assumption 1 *For all $t' \geq t$ there exists some $j \in \mathcal{N}$, such that $\Upsilon_i(t') < \Upsilon_j(t')$.*

Consider that $\ell \in \mathcal{N}$ is set as the leader at time t , and an agent $i' \in \mathcal{N}$ maximizes $\Upsilon_{i'}(t)$ among all agents in \mathcal{N} . Then, from the construction of Alg. 1 and Lemmas 1 and 2, there

exists $\tau_{\ell\nu} \geq t$ when the next leader's desired service $\sigma_{\ell\nu}$ has been provided and a leader re-election is triggered with the time stamp $\tau_{\ell\nu}$. Note that from Eq. (18), $\Upsilon_{i'}(\tau_{\ell\nu})$ is still maximal among the agents in \mathcal{N} , and hence i' becomes the next leader. Furthermore, there exists time $\tau_{i'\nu} \geq \tau_{\ell\nu}$ when the next desired service $\sigma_{i'\nu}$ of agent i' has been provided, and hence $\Upsilon_{i'}(\tau_{i'\nu}) < \Upsilon_j(\tau_{i'\nu})$, for all $j \in \mathcal{N}$, including the agent i .

Since we assume that i does not become a leader for any $t' \geq t$ (Assump. 1), it holds that $\Upsilon_{i'}(t'') < \Upsilon_i(t'')$ for all $t'' \geq \tau_{i'\nu}$. Inductively, we can reason similarly about the remaining agents. As they are only finite number of agents N , after large enough $T \geq t$, we obtain that $\Upsilon_j(t') < \Upsilon_i(t')$ for all j and for all $t' \geq T$. This contradicts Assump. 1 and hence the proof is complete. ■

From Lemmas 1, 2, and 3, the correctness of the high-level plan computation (proven e.g., in [1]), and the construction of Alg. 1, we obtain, that ϕ_i is satisfied for all $i \in \mathcal{N}$.

From Alg. 1, each agent synthesizes its high-level plan Ω_i and waits for the first leader to be elected. Denote the first leader by $\ell_1 \in \mathcal{N}$. Lemma 2 guarantees that there exists a finite time $\bar{t}_1 > 0$ that $x_{\ell_1}(\bar{t}_1) \in R_{\ell_1\nu}$, while at the same time Lemma 1 ensures that the communication network $G(t)$ remains connected, $\forall t \in [0, \bar{t}_1]$. At $\tau_{\ell_1\nu} \geq \bar{t}_1$, the first service $\sigma_{\ell_1\nu}$ of the leader's high-level plan is provided, defining a prefix of the agent i 's trace as $\mathbb{T}_i^A(1) = \tau_{\ell_1\nu}$, $\mathbb{A}_i(\tau_{\ell_1\nu}) = \pi_{i\nu}$, $\mathbb{T}_i^S(1) = \tau_{\ell_1\nu}$, $\mathbb{S}_i(\tau_{\ell_1\nu}) = \sigma_{i\nu}$. Furthermore, $\mathbb{T}_j^A(1) = \tau_{\ell_1\nu}$, $\mathbb{A}_j(\tau_{\ell_1\nu}) = \varpi_{j\nu}$. Afterwards, a new leader $\ell_2 \in \mathcal{N}$ is elected according to Alg. 1 and $R_{\ell_2\nu}$ is set as the goal region. Now the controller from Eq. (7) is switched to the case when ℓ_2 is the leader. By induction, we obtain that for all $t \geq 0$ it holds

- Given a leader ℓ_t and a goal region $R_{\ell_t\nu}$ at time t , there exists $\bar{t} \geq t$, when $x_{\ell}(\bar{t}) \in R_{\ell\nu}$.
- $G(t)$ is connected.

Together with Lemma 3, we conclude that ϕ_i is satisfied for all $i \in \mathcal{N}$.

Corollary 1 *The proposed solution in Alg. 1 is a solution to Problem 1.*

V. EXAMPLE

In the following case study, we present an illustrative example of a team of four autonomous robots with heterogeneous functionalities and capacities. The proposed algorithms are implemented in Python 2.7. All simulations are carried out on a desktop computer (3.06 GHz Duo CPU and 8GB of RAM).

A. System Description

Denote by the four autonomous agents $\mathfrak{R}_1, \mathfrak{R}_2, \mathfrak{R}_3$ and \mathfrak{R}_4 . They all satisfy the dynamics specified by (1). They all have the communication radius $1.5m$, while ε is chosen to be $0.1m$. The workspace of size $4m \times 4m$ is given in Figure 1, within which the regions of interest for \mathfrak{R}_1 are R_{11}, R_{12} (in red), for \mathfrak{R}_2 are R_{21}, R_{22} (in green), for \mathfrak{R}_3 are R_{31}, R_{32} (in blue) and for \mathfrak{R}_4 are R_{41}, R_{42} (in cyan).

Besides the motion among these regions, each agent can provide various services as described in the following: agent \mathfrak{R}_1 can load (l_H, l_A) , carry and unload (u_H, u_A) a heavy object H or a light object A. Besides, it can help agent \mathfrak{R}_4 to assemble (h_C) object C; agent \mathfrak{R}_2 is capable of helping the agent \mathfrak{R}_1 to load the heavy object H (h_H), and to execute two simple tasks (t_1, t_2) without help from others; agent \mathfrak{R}_3 is capable of taking snapshots (s) when being present in its own or others' goal regions; agent \mathfrak{R}_4 can assemble (a_C) object C under the help of agent \mathfrak{R}_1 .

B. Task Description

Each agent within the team is locally-assigned complex tasks that require collaboration: agent \mathfrak{R}_1 has to periodically load the heavy object H at region R_{11} , unload it at region R_{12} , load the light object A at region R_{12} , unload it at region R_{11} . In LTL formula, it is specified as

$$\phi_1 = \text{GF}((l_H \wedge h_H \wedge r_{11}) \wedge \text{X}(u_H \wedge r_{12})) \wedge \text{GF}((l_A \wedge r_{12}) \wedge (u_A \wedge r_{11}));$$

Agent \mathfrak{R}_2 has to service the simple task t_1 at region R_{21} and task t_2 at region R_{22} in sequence, but it requires \mathfrak{R}_2 to witness the execution of task t_2 , by taking a snapshot at the moment of the execution. It is specified as

$$\phi_2 = \text{F}((t_1 \wedge r_{21}) \wedge \text{F}(t_2 \wedge s \wedge r_{22}));$$

Agent \mathfrak{R}_3 has to surveil over both of its goal regions (R_{31}, R_{32}) and take snapshots there, which is specified as

$$\phi_3 = \text{GF}(s \wedge r_{31}) \wedge \text{GF}(s \wedge r_{32});$$

Agent \mathfrak{R}_4 has to assemble object C at its goal regions (R_{41}, R_{42}) infinitely often, which is specified as

$$\phi_4 = \text{GF}(a_C \wedge r_{41}) \wedge \text{GF}(a_C \wedge r_{42}).$$

Note that tasks ϕ_1, ϕ_3 and ϕ_4 require the collaboration task to be performed infinitely often.

C. Simulation Results

Initially, the agents start evenly from the x -axis, i.e., $(0, 0), (1.3, 0), (2.6, 0), (3.9, 0)$. By Definition 5, the initial edge set is $E(0) = \{(1, 2), (2, 3), (3, 4)\}$, yielding a connected $G(0)$.

The system is simulated for 35s, of which the video demonstration can be viewed here [3]. In particular, when the system starts, each agent synthesizes its local plan as described in Section IV-C.1. After running the leader election scheme proposed in Section IV-C, agent \mathfrak{R}_1 is chosen as the leader. As a result, controller (4) is applied for \mathfrak{R}_1 as the leader and the rest as followers, while the next goal region of \mathfrak{R}_1 is R_{11} . As shown by Theorem 2, all agents belong to R_{41} after $t = 3.8s$. After that agent \mathfrak{R}_2 helps agent \mathfrak{R}_1 to load object H. Then agent \mathfrak{R}_2 is elected as the leader after collaboration is done, where R_{21} is chosen as the next goal region. At $t = 6.1s$, all agents converge to R_{21} . Afterwards, the leader and goal region is switched in the following order: \mathfrak{R}_3 as leader to region R_{31} at $t = 6.1s$; \mathfrak{R}_4 as leader to region

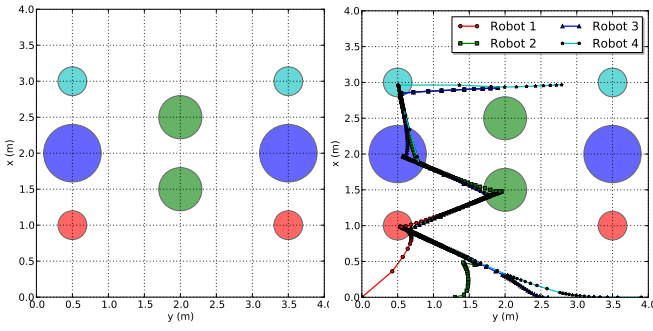


Fig. 1: Left: the workspace structure, where the goal regions for each agent are indicated by color; Right: snapshot of simulation at $t = 11.2s$.

| Time (s) | (0, 3.8) | (3.8, 6.1) | (6.1, 8.1) | (8.1, 10.6) |
|-------------|------------------|------------------|------------------|------------------|
| Leader | \mathfrak{R}_1 | \mathfrak{R}_2 | \mathfrak{R}_3 | \mathfrak{R}_4 |
| Goal Region | R_{11} | R_{21} | R_{31} | R_{41} |
| Time (s) | (10.6, 14.2) | (14.2, 16.3) | (16.3, 18.2) | (18.2, 20.1) |
| Leader | \mathfrak{R}_4 | \mathfrak{R}_2 | \mathfrak{R}_3 | \mathfrak{R}_1 |
| Goal Region | R_{42} | R_{22} | R_{32} | R_{12} |
| Time (s) | (20.1, 24.2) | (24.2, 25.7) | (25.7, 28.1) | (28.1, 31.4) |
| Leader | \mathfrak{R}_1 | \mathfrak{R}_3 | \mathfrak{R}_3 | \mathfrak{R}_4 |
| Goal Region | R_{11} | R_{31} | R_{32} | R_{41} |

TABLE I: Leader Election Scheme

R_{41} at $t = 8.1s$; \mathfrak{R}_4 as leader to region R_{42} at $t = 10.6s$; \mathfrak{R}_2 as leader to region R_{22} at $t = 14.2s$; \mathfrak{R}_3 as leader to region R_{32} at $t = 16.3s$; \mathfrak{R}_1 as leader to region R_{12} at $t = 18.2s$; \mathfrak{R}_1 as leader to region R_{11} at $t = 20.1s$; \mathfrak{R}_3 as leader to region R_{31} at $t = 24.2s$; \mathfrak{R}_3 as leader to region R_{32} at $t = 25.7s$; \mathfrak{R}_4 as leader to region R_{41} at $t = 28.1s$; \mathfrak{R}_4 as leader to region R_{42} at $t = 31.4s$. The above arguments are summarized in Table I.

Figure 1 shows the snapshot of the simulation at time $t = 11.2s$, when agent \mathfrak{R}_4 was chosen as the leader and R_{42} as the goal region. Figure 2 shows the trajectory of $\mathfrak{R}_1, \mathfrak{R}_2, \mathfrak{R}_3, \mathfrak{R}_4$ during time $[0, 34.7s]$, in red, green, blue, cyan respectively. Furthermore, the pairwise distance for neighbours within $E(0)$ is shown in Figure 2. It can be verified that they stay below the constrained radius $1.5m$ thus the agents remain connected.

VI. CONCLUSIONS AND FUTURE WORK

We present a distributed motion and task control framework for multi-agent systems under complex local LTL tasks and connectivity constraints. It is guaranteed that all individual tasks including both local and collaborative services are fulfilled, while at the same time connectivity constraints are satisfied. Further work includes inherently-coupled dynamics and time-varying network topology.

REFERENCES

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
[2] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta. Formal approach to the deployment of distributed robotic teams. *IEEE Transactions on Robotics*, 28(1):158–171, 2012.
[3] Demonstration. <https://www.dropbox.com/s/yhyueagokeihv7k/simulation.avi>.

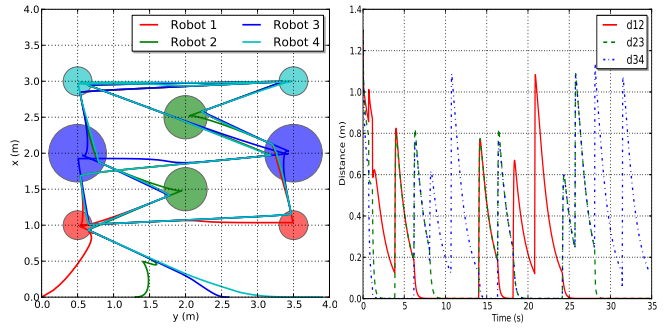


Fig. 2: Left: the agents' trajectory during time $[0, 34.8s]$; Right: the evolution of pair-wise distances $\|x_{12}\|, \|x_{23}\|, \|x_{34}\|$, which all stay below the communication radius $1.5m$ as required by the connectivity constraints.

[4] D.V. Dimarogonas and K.J. Kyriakopoulos. A connection between formation control and flocking behavior in nonholonomic multiagent systems. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 940–945, May 2006.
[5] M. B. Egerstedt and X. Hu. Formation constrained multi-agent control. 2001.
[6] I. Filippidis, D.V. Dimarogonas, and K.J. Kyriakopoulos. Decentralized multi-agent control from local ltl specifications. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 6235–6240, 2012.
[7] H. Garcia-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers*, C-31(1):48–59, 1982.
[8] P. Gastin and D. Oddoux. LTL2BA tool, viewed September 2012. URL: <http://www.lsv.ens-cachan.fr/gastin/ltl2ba/>.
[9] M. Guo and D. V. Dimarogonas. Reconfiguration in motion planning of single- and multi-agent systems under infeasible local LTL specifications. 2013. To appear.
[10] Y. Hong, J. Hu, and L. Gao. Tracking control for multi-agent consensus with an active leader and variable topology. *Automatica*, 42(7):1177–1182, 2006.
[11] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.
[12] S. Karaman and E. Frazzoli. Vehicle routing with temporal logic specifications: Applications to multi-UAV mission planning. *International Journal of Robust and Nonlinear Control*, 21:1372–1395, 2011.
[13] H. K. Khalil and J. W. Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, 2002.
[14] M. Kloetzer, X. C. Ding, and C. Belta. Multi-robot deployment from ltl specifications with reduced communication. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference*, pages 4867–4872, 2011.
[15] S. G. Loizou and K. J. Kyriakopoulos. Automated planning of motion tasks for multi-robot systems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, volume 44, December 2005.
[16] M. M. Quottrup, T. Bak, and R. I. Zamanabadi. Multi-robot planning: a timed automata approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 4417–4422, 2004.
[17] W. Ren. Multi-vehicle consensus with a time-varying reference state. *Systems & Control Letters*, 56(7):474–483, 2007.
[18] W. Ren, R. W. Beard, and E. M. Atkins. A survey of consensus problems in multi-agent coordination. In *American Control Conference, 2005. Proceedings of the 2005*, pages 1859–1864 vol. 3, June 2005.
[19] J. Tumova and D. Dimarogonas. A receding horizon approach to multi-agent planning from ltl specifications. In *Proceedings of the American Control Conference (ACC)*, 2014. To appear.
[20] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus. Optimality and robustness in multi-robot path planning with temporal logic constraints. *International Journal of Robotics Research*, 32(8):889–911, 2013.
[21] C. Wiltse, F. A. Ramponi, and J. Lygeros. Synthesis of an asynchronous communication protocol for search and rescue robots. pages 1256–1261, 2013.